**Generics – Part\_04**

* **Generic methods and wild-card character (?)**

1. m1(ArrayList<String> l)

We can call this method by passing, ArrayList of only String type.

But within the method we can add only String type of Objects to the list. By mistake, if we are trying to add any other type then we will get compile-time error.

m1(ArrayList<String> l){

l.add(“A”); // Valid

l.add(null); // Valid

l.add(10); // Invalid - CE

}

1. m1(ArrayList<?> l)

We can call this method by passing, ArrayList of any type.

But within a method we can’t add anything to the List, except null. Because, we don’t know the type exactly.

Null is allowed, because it is valid value for any type.

m1(ArrayList<?> l){

l.add(10.5); // Invalid

l.add(“A”); // Invalid

l.add(10); // Invalid

l.add(null);

}

This type methods are best suitable for read-only operation.

1. m1(ArrayList<? Extends X> l)

X can be either class or interface.

If X is a class, then we can call this method by passing ArrayList of either X type or its child classes.

If X is an interface then we can call this by passing ArrayList of either X type or its implementation classes.

But within a method, we can’t add anything to the List except null, because we don’t know the type of X exactly.

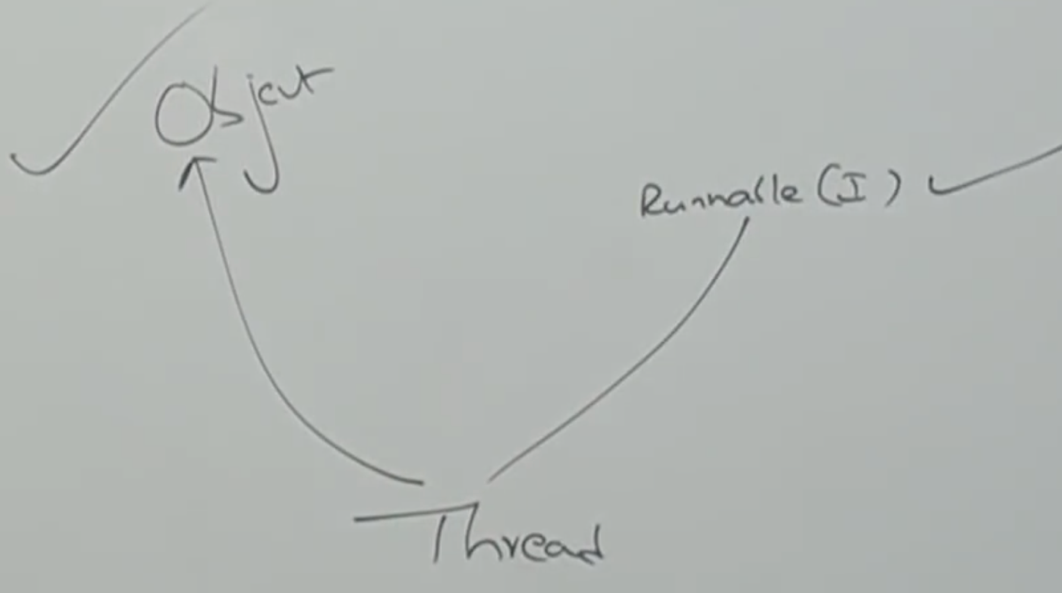
This type of methods also best suitable for read-only operation.

1. m1(ArrayList<? super X> l)

X can be either class or interface.

If X is a class then we can call this method by passing ArrayList of either X type or its super classes.

If X is an interface then we can call this method by passing ArrayList of either X type or super class of implementation class of X.



But within a method we can add X type of objects and null to the List.

* **Examples:**

ArratList<String> l = new ArrayList<String>(); // Valid

ArrayList<?> l = new ArrayList<String>(); // Valid

ArrayList<?> l = new ArrayList<Integer>(); // Valid

ArrayList<? extends Number> l = new ArrayList<Integer>(); // Valid

ArrayList<? extends Number> l = new ArrayList<String>(); // Invalid

CE: incompatible types

found: ArrayList<String>

required: ArrayList< ? extends Number>

ArrayList<? super String> l = new ArrayList<Object>(); // Valid

ArrayList<?> l = new ArrayList<?>(); // Invalid

CE: unexpected type

found: ?

required: class or interface without bounds

ArrayList<?> l = new ArrayList<? extends Number>(); // Valid

CE: unexpected type

found: ? extends Number

required: class or interface without bounds.

Note:

In the right-hand side, we shouldn’t use the wildcard character, we have to define always the specific type. Otherwise we will get compile time error.